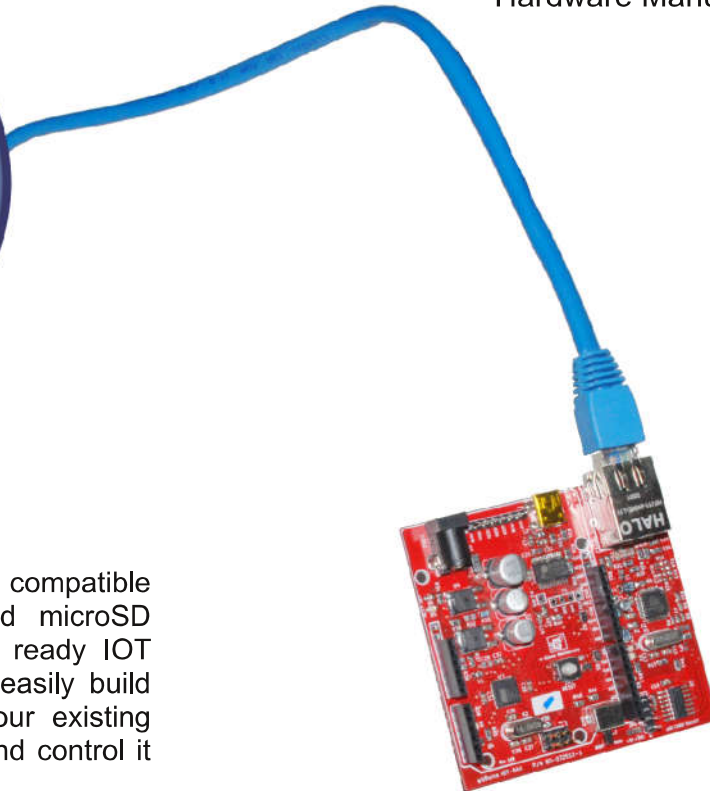


gizDuino IOT-644

Internet Ready Arduino Compatible

Hardware Manual Rev 1R0



gizDuino IOT644 is an Arduino compatible platform with on-board Ethernet and microSD interface circuitry, primed for internet ready IOT projects. With this platform, you can easily build hardware that you can plug into your existing LAN network, and remotely operate and control it anywhere in the world via the internet.

ATMEGA644P Microcontroller

The main controller engine is a ATMEGA644P microcontroller, a close kin, but more feature laden, of the ATMEGA328 used in the Arduino Uno, gizDuino 328, and compatible boards. The ATMEGA644 has twice the program memory (64Kbytes), twice the RAM (4Kbytes), and twice the EEPROM space (2Kbytes) of the ATMEGA328 based boards. This allows user to create codes that can do pretty much more complex features and operation without quickly running out of memory space.

In addition, the ATMEGA644 has an additional hardware USART Serial1, and ten more additional DIO, eight of which are available to user applications. Two of extended DIO pins are for the dedicated use to select CS the on-board Ethernet Controller W5500 and microSD card reader circuitry, instead of the 'usual' DIO10 used by most shield to interface with the SPI. This keeps this pin free to make the gizDuino IOT-644 ready to accept additional shield that uses the SPI.

Features:

Platform:

MCU: ATmega644P @ 16MHz
Flash Memory: 64KB (2x)
RAM: 4KB (2x)
EEPROM: 2KB (2x)
USART: 2 (2x)
I/O: 30 (10 more)

Ethernet:

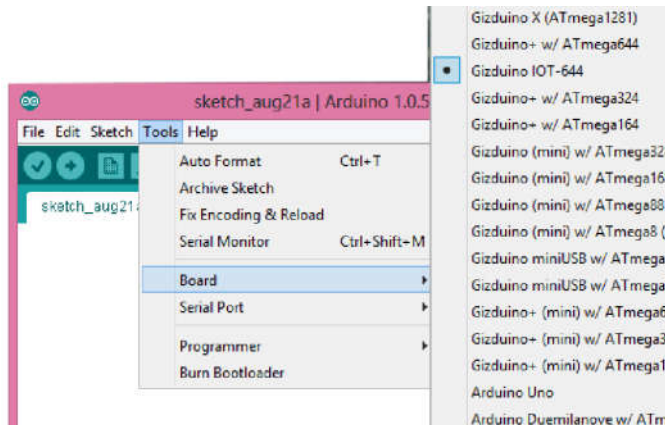
PHY: Wiznet W5500
Speed: 10/100 Mbps
Hardwired TCP/IP : TCP, UDP,
IPv4, ICMP, ARP, IGMP

MicroSD Socket
Bootload Switch
DC Jack
Port for Wireless Module

Read Me First

Board Definition

Check if your Arduino IDE is ready for your gizDuino IOT644 by clicking Tools>Board. You should see gizDuino IOT-644 included in the displayed list. If not, you have to download and install the patches as described in sectionxx.



Arduino Sketch Compatibility

In as far as the user is concerned, the only difference between an Arduino and gizDuino equivalent platform is the USB driver. Once the correct PL2303 USB driver is installed in your PC (you do this only on first time install), you program the gizDuino IOT-644 as you would an Arduino. Sketches and libraries written for Arduino UNO/Diecimilla will generally run with the gizDuino IOT-644 without modification.

However, sketches and libraries containing code that directly access microcontroller features may require rewriting of some portion of the codes. Coding using direct port/feature access is discouraged by the Arduino programming paradigm to ensure compatibility with the widest possible hardware platform, but it cannot stop some programmers from doing so, hence in some but only very few cases, compatibility issues may occur.

Reserved DIOs

The gizDuino IOT-644 uses the SPI PORT to communicate with the on-board ethernet controller and microSD.

These restrictions apply only when you use these on-board peripherals.

The SPI interface share pins with:

DIO11
DIO12
DIO13

Hence, the mentioned pins must not be used for any other purpose other than an SPI communication interface when the onboard ethernet or microSD card port is used.

DIO10 (SPI SS pin) is used mostly by add-ons shield that requires the SPI channel. This pin is available for you add-ons.

Ethernet and SD card Chip Select CS

To keep DIO10 free and available for your add-ons shield, the on board Ethernet and SD card circuitry used pins from the extended DIO. These calls for a slight adjustment in the application code as follows:

SD Library

Fortunately, the SD Library allows easily remapping of the CS pin using the SD.begin() library function.

Simply use (or replace) as follows:

```
SD.begin(27);
```

This basically informs the SD library that the DIO27 is used as the hardware Chip Select CS line. This function is normally put (or can be found for modification) within the void setup() function.

W5500 Library

The W5500 ethernet library CS pin is hardcoded within the library with DIO10, hence, modifying the code is not as easy and straightforward as the SD library.

You have to find and modify the lines of codes that defines SS pin usage within W5500.h library header file:

```
...inline static void initSS()...{ DDRC |= _BV(2); };  
...inline static void setSS()...{ PORTC &= ~_BV(2); };  
...inline static void resetSS()...{ PORTC |= _BV(2); };
```

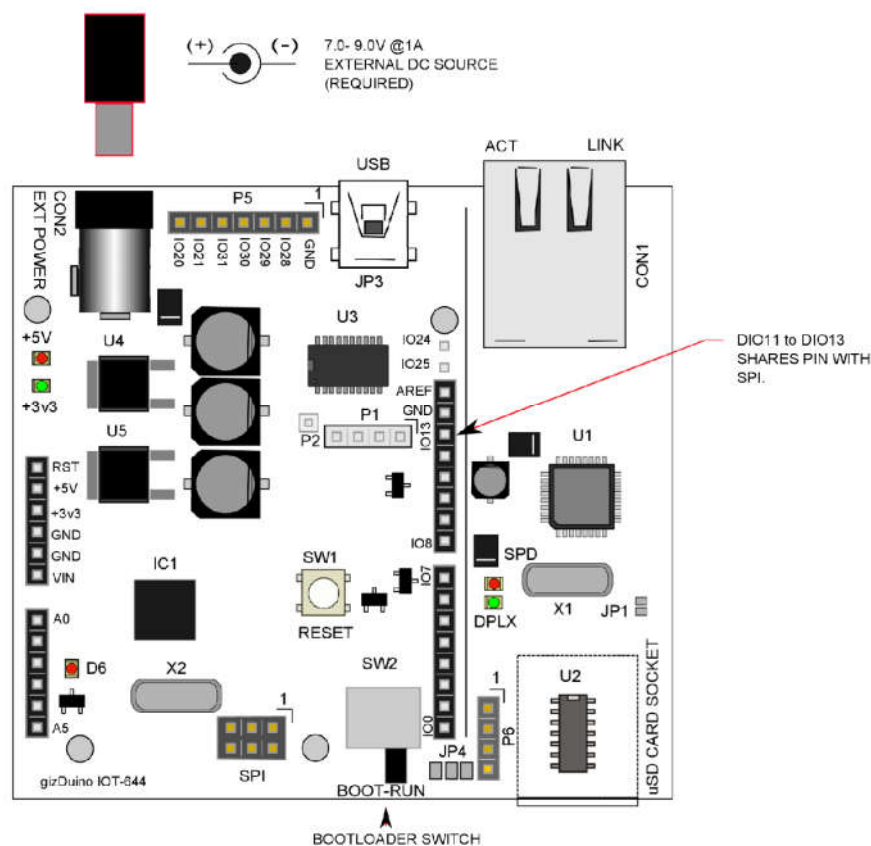


Figure 1. Keep in mind that 1) External DC power source is required. 2) DIO11-DIO13 are used by the SPI channel and must not be used for any other purpose if the on-board ethernet or microSD card interface is used. 3) BOOT switch must be switched to BOOT position during program transfers from PC Arduino IDE. You can leave the switch to BOOT position if no add-on circuit or shield is using IO0- IO1/ UART port.

A W5500.h copy with the mentioned modifications can simply be downloaded from this link. Simply replace the existing W5500.h found on libraries>Ethernet>utility with the downloaded file of the same name. Of course, you need not do this if you installed your Arduino IDE using the latest package supplied by e-Gizmo.

Power Supply

The gizDuino IOT-644 requires an external power to function. The IOT-644 requires a fair amount of current to operate. Although a typical USB port can still supply this amount of current, an external power will minimize the possibility of overloading your PC USB port, ensures enough power to circuitry you may add, keeps you PC USB port less prone to damage from silly things you may do during your experimentations.

DC power source with 7V-10V output with 1A or greater capacity is recommended.

Bootloader Switch

The USB to serial interface circuitry may share the microcontroller USART port (DIO0 and DIO1) with other shields or circuitry. This usually results in UART bus contention that may cause the gizDuino failing to connect with the PC Arduino IDE. Previous generations of gizDuinos (and Arduinos) requires you the remove the contending shield or circuitry during bootloading to ensure a successful link with the PC IDE. With the bootloader switch, physically removing the shields (or circuitry) becomes unnecessary. Just slide the switch to BOOT position during the bootloading process, and then slide it back to RUN thereafter.

You can just leave the switch to BOOT position if you don't have any shield/circuitry installed that uses a connection to the DIO0 and DIO1 UART port pins.

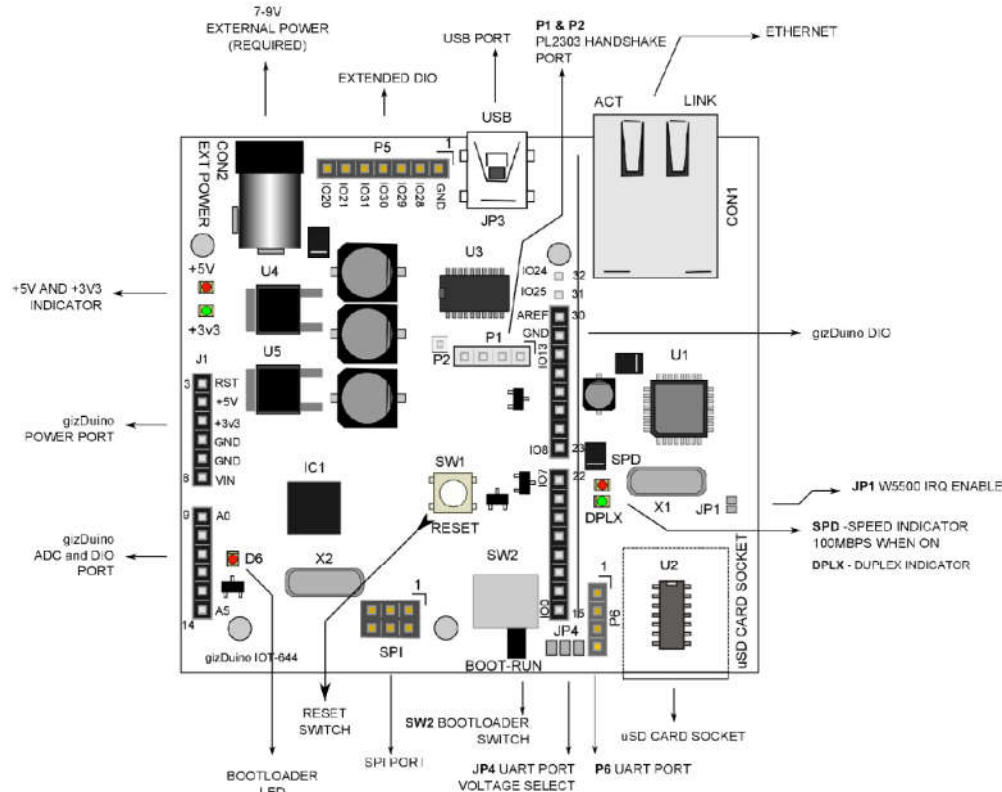


Figure 2. Location guide of major components, terminals, and indicators.

Table 1. CON2 POWER

DC Power Supply Input with reverse polarity diode protection. Required for operation.

Table 2. P5 Extended I/O

Additional I/O available for user applications.

PIN	ID	DESCRIPTION
1	GND	Common GND
2	IO28	DIO, PC4/TD0
3	IO29	DIO, PC5/TDI
4	IO30	DIO, PC6
5	IO31	DIO, PC7
6	IO21	DIO, PA7/ADC7
7	IO20	DIO, PA6/ADC6

Table 3. P1 & P2 Handshake Pins

PL2303 USB to UART converter communications handshake lines. All I/Os listed in the table are active lows.

PIN	ID	DESCRIPTION
P1		
1	CTS	INPUT, Clear To Send
2	DCD	INPUT, Data Carrier Detect
3	DSR	INPUT, Data Set Ready
4	RI	INPUT, Ring Indicator
P2		
1	RTS	OUTPUT, Request To Send

Table 4. J1 gizDuino/Arduino Port

Arduino pinout for shields and expansion modules.

PIN	ID	DESCRIPTION
3	M_RST	INPUT, Reset
4	+5V	POWER OUT, +5V
5	+3V3	POWER OUT, +3V3
6	GND	POWER GND
7	GND	POWER GND
8	+VIN	POWER IN, 7-9VDC
9	AD0	ADC IN0, DIO14
10	AD1	ADC IN1, DIO15
11	AD2	ADC IN2, DIO16
12	AD3	ADC IN3, DIO17
13	AD4	ADC IN4, DIO18
14	AD5	ADC IN5, DIO19
15	IO0	DIO0, USART RX
16	IO1	DIO1, USART TX
17	IO2	DIO2, INT0/USART1 RX
18	IO3	DIO3, INT1/USART1 TX
19	IO4	DIO4, OC1B
20	IO5	DIO5, OC1A
21	IO6	DIO6, OC2B/ICP
22	IO7	DIO7, OC2A
23	IO8	DIO8, AIN0/INT2
24	IO9	DIO9, AIN1/OC0
25	IO10	DIO10, OCOB/SS
26	IO11	DIO11, MOSI
27	IO12	DIO12, MISO
28	IO13	DIO13, SCK
29	GND	GND
30	AREF	ANALOG REFERENCE
31	IO25	DIO25, SDA (EXTENDED IO)
32	IO24	DIO24, SCK (EXTENDED IO)

Table 5. P6 Powered UART PORT

UART port with voltage selcetable power pins for quick connection with e-Gizmo wireless modules. IO0 RX and IO1 TX pin are shared with this port.

PIN	ID	DESCRIPTION
1	IO1	DIO1, USART TX
2	IO0	DIO0, USART RX
3	GND	GND
4	+V	+5V/+3V3 (SEE TABLE 6)

Table 6. JP4 Voltage Selector for P6

P6 +V voltage can be set to +5V or +3V3 by shorting JP4 pad2 with either the +5V or +3V3 pad.

Warning: Do not short with BOTH +5V and +3V3. This will result in a permanent damage to the circuit board.

PIN	ID	DESCRIPTION
1	+3V3	+3V3 PAD
2	C	JP4 COMMON PAD
3	+5V	+5V PAD

Table 7. Serial Peripheral Interface SPI and Programming Port

PIN	ID	DESCRIPTION
1	GND	Common GND
2	IO28	DIO, PC4/TD0
3	IO29	DIO, PC5/TDI
4	IO30	DIO, PC6
5	IO31	DIO, PC7
6	IO21	DIO, PA7/ADC7

PCBs and SCHEMATIC

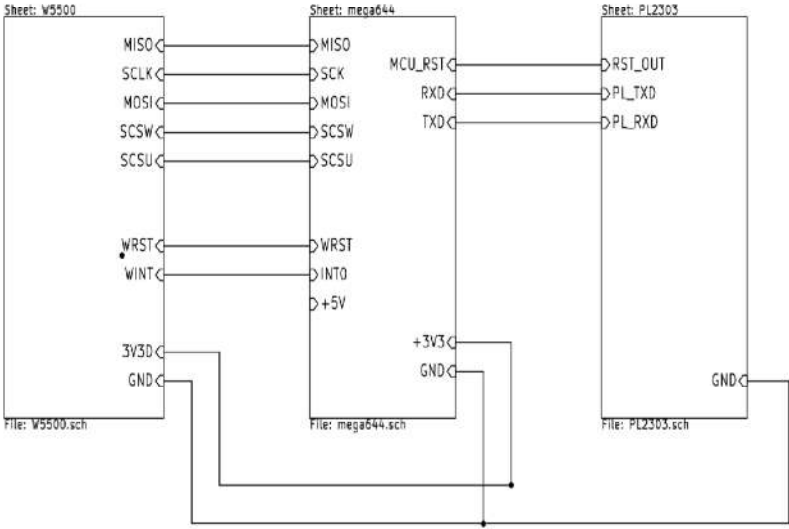


Figure 3. Wiring Block Diagram of gizDuino IOT-644.

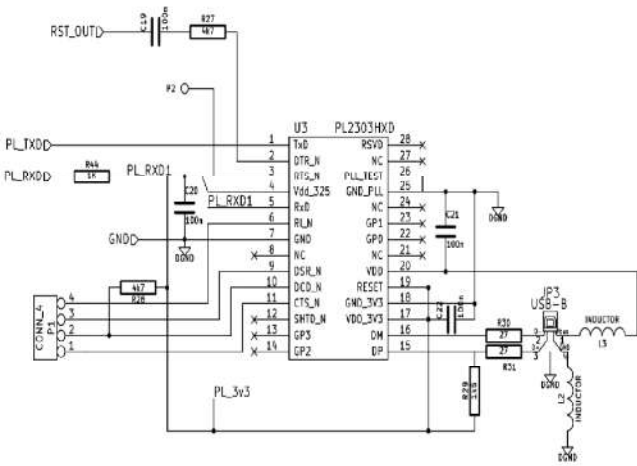


Figure 4. PL2303 USB to UART converter schematic.

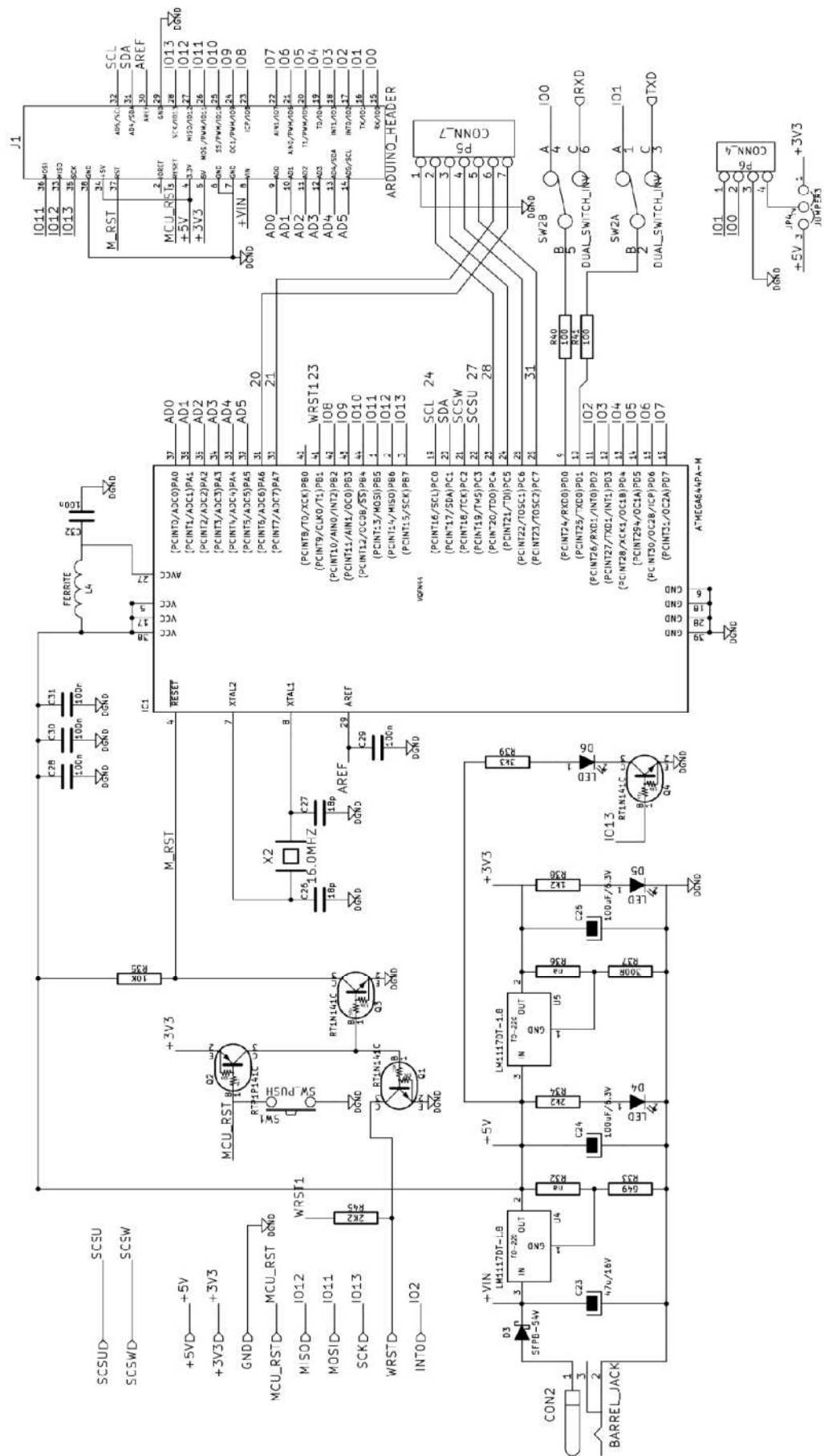


Figure 5. gizDuino IOT-644 microcontroller block schematic diagram.

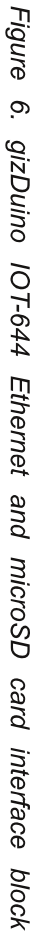


Figure 7. gizDuino IOT-644 component layout guide.

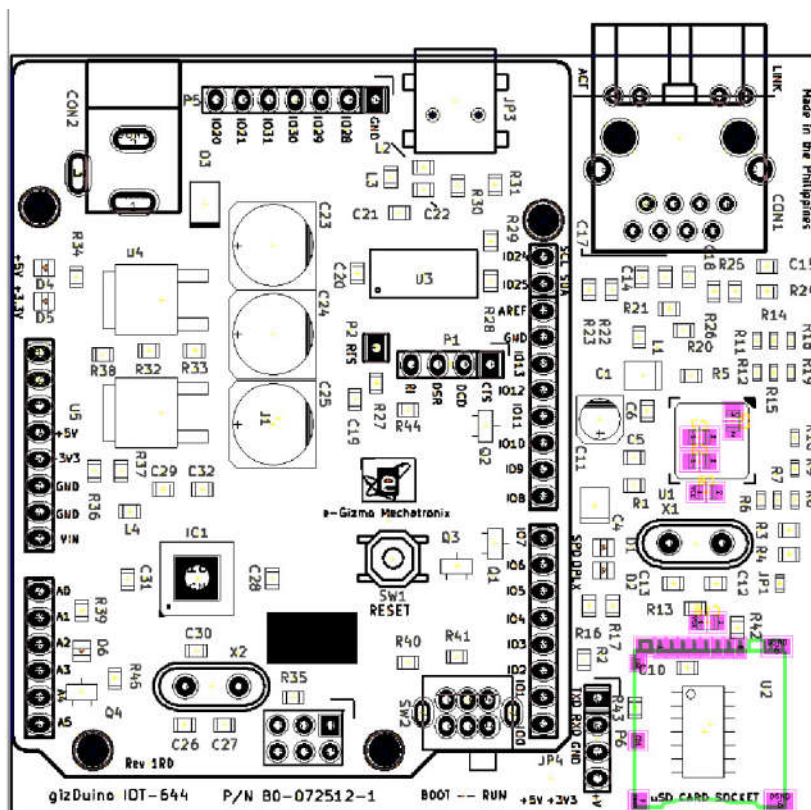


Figure 8. gizDuino IOT-644 component side copper trace guide.

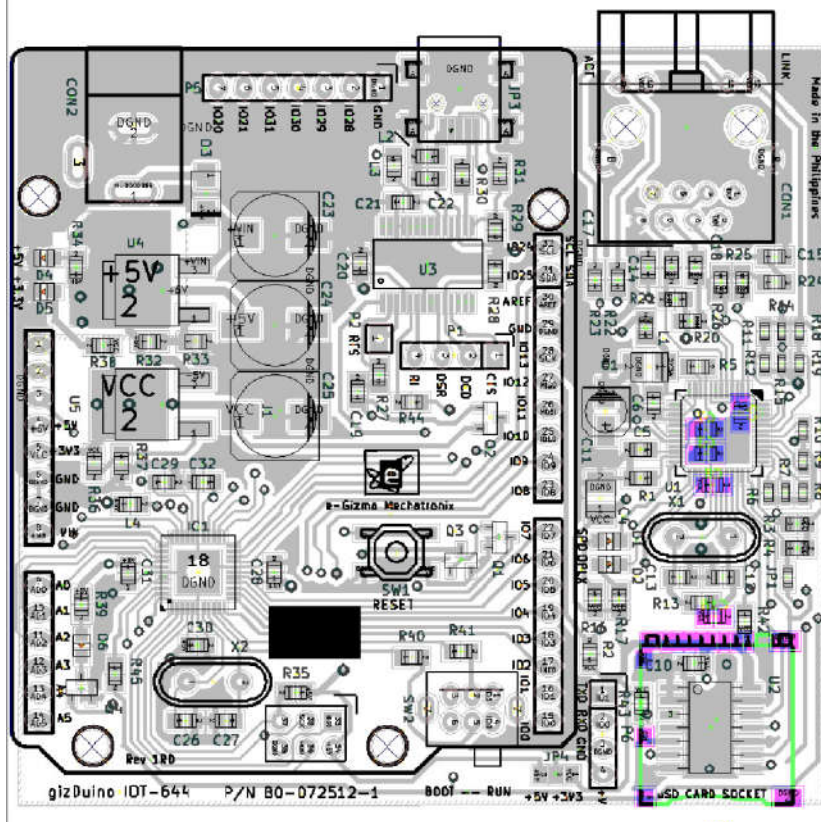


Figure 9. gizDuino IOT-644 bottom side copper trace guide.

